

CONFIDENTIAL

ATTORNEY'S DOCKET NO. MS1-535US

1 This application claims priority to a provisional patent application No.
2 60/192,170, entitled "DLL/COM Redirection", filed on 3/27/00 by the inventors of
3 this application.

4 5 **TECHNICAL FIELD**

6 This invention relates to computer application programs and, more
7 particularly, to DLL/COM (dynamic-link library/component object model)
8 redirection in computer application programs that utilize shared components.

9 10 **BACKGROUND**

11 Modern operating systems and applications are built from many
12 components. A component is a self-contained software entity, offering a set of
13 functions that can be used broadly by a variety of applications. A component is
14 typically code, but it may also include the registry state, support files, etc.
15 Component sharing enables efficiencies since individual components are used by
16 more than one application. This enables efficiencies attributable to not having to
17 re-write and test redundant code. This allows a developer to distribute a product
18 to market more quickly and provides more stable code (when done properly).

19 Successful global component sharing requires that any shared component
20 function exactly like previous versions of that component. In practice, however,
21 100 percent backward compatibility is difficult if not impossible to achieve
22 because of the difficulty in testing all configurations in which the shared
23 component may be used. Both newer and older applications end up using the
24 same component, and over time, fixing and improving the component becomes
25 increasingly difficult.

1 interoperability between applications produced by multiple software vendors,
2 bringing cost reduction and heightened software efficiency.

3 However, there are costs to sharing. Sharing means that applications
4 become interdependent upon one other, introducing an element of fragility.
5 Changes to one component may produce unintended effects in other components.
6 Typically, an application may become dependent on a particular version of a
7 shared component. Another application may be installed with an upgraded (or
8 downgraded) version of that shared component, and the first application may
9 suffer from that change. In the extreme, applications that once worked well
10 mysteriously start to function oddly, or even fail. This condition is often referred
11 to by developers as "DLL Hell."

12 Isolation

13 The opposite of sharing in a system is isolation. Applications can be
14 isolated by statically binding all resources and code into the application.
15 However, complete isolation is not feasible today for applications that rely on
16 COM components or any other globally stored system resources.

17 One solution discussed herein for reducing application fragility is to
18 selectively isolate applications and components. Under this scenario, applications
19 may all have access to the same component, but multiple versions of that
20 component now become available. Component producers have the freedom to
21 produce new versions of old components, making improvements and fixing bugs.
22 Customers, on the other hand, can choose the version that fits with a particular
23 application.

24 With components, the key is to provide the version that is appropriate to
25 each application and isolate the different versions from each other. Further, with

redirection, applications can be configured to use the component version that fits with that particular application, regardless of what other versions are currently deployed or will be deployed in the future. In addition, a developer may apply a bug fix to a component in the context of a single application without having to be concerned with the effect of the bug fix on other applications.

SUMMARY

The described implementations contemplate a new form of component sharing called side-by-side sharing, which uses selective isolation to minimize the problems associated with “DLL Hell.” Side-by-side sharing allows multiple versions of the same component to run at the same time in different processes. Applications can then use a specific version of a component for which they were designed and tested, even if another application requires a different version of the same component. This arrangement allows developers to build and deploy more reliable applications because developers are able to specify the version of the component they will use for their application, independent of the other applications on the system.

It is noted that multiple versions of a side-by-side DLL may also be used in the same process. It is possible in an atypical case that different versions cannot be loaded at the same time because of the versions of the components were not designed to be truly side-by-side. Even then, this implementation provides value as different component versions may still be used in different processes albeit mutually exclusively.

A specific implementation described herein is DLL/COM redirection. Using this strategy, developers and administrators repackage existing applications

1 **BRIEF DESCRIPTION OF THE DRAWINGS**

2 A more complete understanding of the various methods and arrangements
3 of the present invention may be had by reference to the following detailed
4 description when taken in conjunction with the accompanying drawings, wherein:

5 Fig. 1 is a block diagram generally illustrating a computer system that is
6 suitable for use with the described implementations.

7 Fig. 2 is a flow diagram of a method for utilizing DLL/COM redirection in
8 accordance with one described implementation.

9
10 **DETAILED DESCRIPTION**

11 In the described implementations and illustrations, wherein like reference
12 numerals refer to like elements, as being implemented in a suitable computing
13 environment. Although not required, the invention will be described in the general
14 context of computer-executable instructions, such as program modules, being
15 executed by a personal computer. Generally, program modules include routines,
16 programs, objects, components, data structures, etc. that perform particular tasks
17 or implement particular abstract data types. Moreover, those skilled in the art will
18 appreciate that the invention may be practiced with other computer system
19 configurations, including hand-held devices, multi-processor systems,
20 microprocessor based or programmable consumer electronics, network PCs,
21 minicomputers, mainframe computers, and the like. The invention may also be
22 practiced in distributed computing environments where tasks are performed by
23 remote processing devices that are linked through a communications network. In
24 a distributed computing environment, program modules may be located in both
25 local and remote memory storage devices.

environment, such registration requires that while the implementation of the DLL/COM component can change between versions, such registered COM meta-data as CLSID, ProgID, Type Library, and Threading Model cannot change between versions.

Using DLL/COM Redirection

DLL/COM redirection allows a developer or administrator to selectively isolate existing components to the application they are building and deploying. This section discusses how to activate DLL/COM redirection, and how to select which components to isolate.

DLL/COM redirection is activated on an application-by-application basis by the presence of a ".local" file, which is referenced by the local file pointer 146 stored in directory 124. The ".local" file is an empty file in the same directory as the application's executable (.exe) file, with the same name as the application's executable file with ".local" appended to the end of the name.

For example, to activate DLL/COM redirection for an application called "myapp.exe," an empty file named "myapp.exe.local" is created and stored in the same directory where myapp.exe is installed, *e.g.*, a pointer referencing myapp.exe.local is stored in the same directory as a pointer referencing myapp.exe.

Once DLL/COM redirection is activated, whenever the application loads a DLL or a COM component (.OCX file), the operating system 121 looks first for a pointer to the DLL or OCX in the same directory where a pointer to the executable file of the application is installed. If such a file is found, the operating system 121 then looks for a local version of a DLL/COM component. If a version of the DLL

